February 1992

UILU-ENG-92-2209
CRHC-92-06

Center for Reliable and High-Performance Computing

Do NOT Remove

REDUCING SPACE OVERHEAD FOR INDEPENDENT CHECKPOINTING

**Y.-M. Wang, P.-Y. Chung,
I.-J. Lin, and W. K. Fuchs**

Coordinated Science Laboratory
College of Engineering
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Approved for Public Release. Distribution Unlimited.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-92-2209 CRHC-92-06	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois	6b. OFFICE SYMBOL (If applicable) N/A
7a. NAME OF MONITORING ORGANIZATION NASA ICLASS and JSEP	
6c. ADDRESS (City, State, and ZIP Code) 1101 W. Springfield Avenue Urbana, IL 61801	7b. ADDRESS (City, State, and ZIP Code) MoffittField, CA Washington, DC Washington, DC
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 7a	8b. OFFICE SYMBOL (If applicable)
9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS (City, State, and ZIP Code) 7b	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

Reducing Space Overhead for Independent Checkpointing

12. PERSONAL AUTHOR(S)

Wang, Yi-Min; Pi-Yu Chung; In-Jen Lin; and W. Kent Fuchs

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM _____ TO _____	92-03-11	32

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	recovery line, algorithm, non-discardable checkpoints, communication-trace-driven simulation

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The main disadvantages of independent checkpointing are the possible domino effect and the associated storage space overhead for maintaining multiple checkpoints. In most previous work, it has been assumed that only the checkpoints older than the current global recovery line can be discarded. In this paper, we generalize the notion of recovery line to potential recovery line. Only the checkpoints belonging to at least one of the potential recovery lines can not be discarded. By using the model of maximum-sized antichains on a partially ordered set, an efficient algorithm is developed for finding all non-discardable checkpoints and we show that the number of non-discardable checkpoints can not exceed N(N+1)/2 where N is the number of processors. Communication-trace-driven simulation for several hypercube programs is performed to show the benifit of the proposed algorithm for real applications.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

REDUCING SPACE OVERHEAD FOR INDEPENDENT CHECKPOINTING [1]

Yi-Min Wang, Pi-Yu Chung[†], In-Jen Lin[‡] and W. Kent Fuchs

Center for Reliable and High-Performance Computing
University of Illinois

[†] Coordinated Science Laboratory
University of Illinois

[‡] Department of Mathematics
University of Illinois

Primary contact: Yi-Min Wang

Center for Reliable and High-Performance Computing
Coordinated Science Laboratory
1101 W. Springfield Ave.
University of Illinois
Urbana, IL 61801

E-mail: ymwang@crhc.uiuc.edu
Phone: (217) 244-7161
FAX: (217) 244-5686

ABSTRACT

The main disadvantages of independent checkpointing are the possible domino effect and the associated storage space overhead for maintaining multiple checkpoints. In most previous work, it has been assumed that only the checkpoints older than the current global recovery line can be discarded. In this paper, we generalize the notion of recovery line to potential recovery line. Only the checkpoints belonging to at least one of the potential recovery lines can not be discarded. By using the model of maximum-sized antichains on a partially ordered set, an efficient algorithm is developed for finding all non-discardable checkpoints and we show that the number of non-discardable checkpoints can not exceed $N(N+1)/2$ where N

[1] *Acknowledgement:* This research was supported in part by the National Aeronautics and Space Administration (NASA) under Grant NASA NAG 1-613, in cooperation with the Illinois Computer Laboratory for Aerospace Systems and Software (ICLASS), and in part by the Department of the Navy and managed by the Office of the Chief of Naval Research under Contract N00014-91-J-1283.

is the number of processors. Communication-trace-driven simulation for several hypercube programs is performed to show the benifit of the proposed algorithm for real applications.

Key words: fault tolerance, message-passing systems, independent checkpointing, recovery lines, checkpoint space reclamation

I INTRODUCTION

Numerous checkpointing and rollback recovery techniques have been proposed in the literature for parallel systems. They can be classified into two categories. *Coordinated checkpointing* schemes synchronize computation with checkpointing by coordinating processors during a checkpointing session in order to maintain a consistent set of checkpoints [1, 2, 3]. Each processor only keeps the most recent checkpoint and rollback propagation is avoided at the cost of potentially significant performance degradation during normal execution. *Independent checkpointing* schemes replace the above synchronization by dependency tracking and possibly message logging [4, 5, 6, 7] in order to preserve process autonomy. Possible rollback propagation in case of a fault is handled by reconstruction of a consistent system state based on the dependency information. Lower run-time overhead during normal execution is achieved by maintaining multiple checkpoints and allowing slower recovery.

This paper considers the independent checkpointing schemes. Most research on this subject has concentrated on algorithms for finding the latest consistent set of checkpoints, i.e., the *recovery line*, during rollback recovery. The same algorithms can be applied to the set of existing checkpoints during normal execution to find the *current global recovery line*. All the checkpoints older than the current global recovery line then become obsolete checkpoints and therefore can be discarded. When the domino effect [3, 8] occurs, large number of non-obsolete checkpoints have to be kept on the stable storage and result in large space overhead.

Our approach is based on the observation that many non-obsolete checkpoints can also be discarded because they will never become members of any future recovery line. The notion of recovery line is generalized to *potential recovery line*. A checkpoint is non-discardable if and only if it belongs to at least one of the potential recovery lines. By modeling a recovery line as the maximum maximum-sized antichain on a partially ordered set, an efficient algorithm

is presented for finding the union of all potential recovery lines, which gives the set of non-discardable checkpoints. A maximum on the size of this set is also derived to show that even when domino effect persists during program execution, the space overhead for maintaining multiple checkpoints will not grow without limit.

The outline of the paper is as follows. Section II describes the system model; background materials are introduced in Section III; Section IV formulates the problem; Section V gives the necessary and sufficient conditions for a checkpoint to be non-discardable; the checkpoint space reclamation algorithm is developed in Section VI; the maximum number of non-discardable checkpoints is derived in Section VII; Section VIII extends the results to recovery protocols using virtual checkpoints and Section IX is the conclusion.

II SYSTEM MODEL

A Checkpointing and Rollback Recovery

The system model considered in this paper is a message-passing system consisting of a number of concurrent processes for which all process communication is through message passing. Processes are assumed to run on fail-stop processors [9] and each processor is considered an individual recovery unit.

During normal execution, the state of each processor is occasionally saved as a *checkpoint* on stable storage and can be reloaded for rollback recovery in case of a detected error. Let CP_{ik} denote the kth checkpoint of processor p_i with $k \geq 0$ and $0 \leq i \leq N-1$, where N is the number of processors. A *checkpoint interval* is defined to be the time between two consecutive checkpoints on the same processor. Each processor takes its checkpoint independently, i.e. without synchronizing with any other processors, and includes in each

checkpoint the *communication information* containing:

1. its own processor number and checkpoint number and
2. the sender's processor number and checkpoint number tagged on each message it has received during the previous checkpoint interval.

A centralized *checkpoint space reclamation algorithm* can be invoked by any processor occasionally to collect the global communication information, construct the dependency graph, determine the set of obsolete checkpoints and reclaim the storage space.

We consider two different rollback recovery procedures, Schemes A and B. Scheme A basically follows the algorithm described by Bhargava and Lian [6] and is summarized as follows. When a processor p_i detects an error, it starts a two-phase centralized recovery procedure. First, a *rollback-initiating* message is sent to every other processor to request the up-to-date communication information. Each surviving processor takes a *virtual checkpoint* upon receiving the *rollback-initiating* message so that the communication information during the most recent checkpoint interval is also collected. After receiving the responses, p_i constructs the complete dependency graph and executes the *rollback propagation algorithm* (described in the next section) to determine the recovery line. A *rollback-request* message is then sent to each processor. The message requests each involved processor to reload the checkpoint in the recovery line and restart.

In this paper, Scheme B is proposed as a variation of Scheme A. Instead of a virtual checkpoint, a real checkpoint is taken by each surviving processor upon receiving the *rollback-initiating* message. The recovery line then consists of all real checkpoints. This modified scheme takes advantage of the coordination needed for recovery and can often force the current recovery line to move forward. Each processor can then discard all checkpoints except the one belonging to this recovery line. *Rollback propagation* for possible recovery in

the future is therefore bounded by this new recovery line.

B Consistency of Checkpoints

There are two situations concerning the consistency between two checkpoints. In Fig. 1(a), if p_i and p_j restart from the checkpoints CP_{ik} and CP_{jm} respectively, the message m is recorded as "received but not yet sent". In a general model without the assumption of deterministic execution, message m is not guaranteed to be re-sent during reexecution. CP_{ik} and CP_{jm} are thus inconsistent.

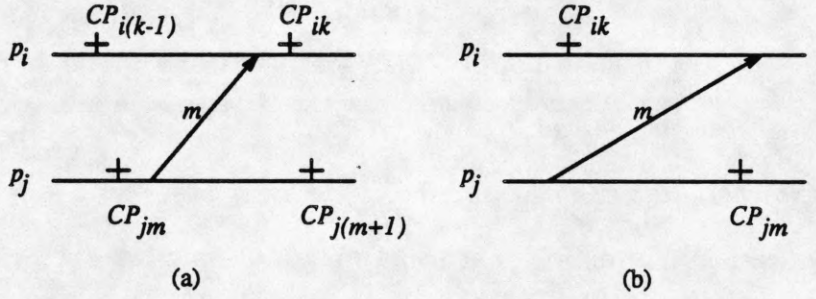


Figure 1: (a) Inconsistent checkpoints; (b) consistent checkpoints.

Fig. 1(b) illustrates the second situation. The message m is recorded as "sent but not yet received" according to the system state containing CP_{ik} and CP_{jm}. By defining the *state of the channels* to be the set of messages sent but not yet received, it has been proved [2, 10] that checkpoints like CP_{ik} and CP_{jm} can be considered consistent if the corresponding state of the channels is also recorded. In Koo and Toueg's paper [3], such state was assumed to be recorded at the sender side in the form of lost messages and the set of messages was guaranteed to be delivered reliably by some end-to-end transmission protocol. When it is impossible or difficult to implement the above scheme, pessimistic message logging [11, 12, 13] can ensure the state of the channels is properly recorded at the receiving end. As a result, we consider the situation in Fig. 1(b) as consistent.

III PRELIMINARIES

A Partially Ordered Set and Checkpoint Graph

In a message-passing system, an event *a happens before* event *b* [14] if and only if

1. *a* and *b* are events in the same processor, and *a* occurs before *b*; or
2. *a* is the sending of a message by one processor and *b* is the receiving of the same message by another processor; or
3. *a* happens before *c* and *c* happens before *b*.

The set of events with the "happens before" relation forms a *partially ordered set*, or *poset* [14]. When dealing with the problem of finding a consistent set of checkpoints, we only consider the *induced subposet* [15] $P = (C, <)$, where C is the set of all checkpoints and $<$ is the "happens before" relation.

A *checkpoint graph (CPG)*, of which the transitive closure is the poset P, is a directed acyclic graph constructed as follows [4]. Each vertex on the checkpoint graph represents a checkpoint. A directed edge exists from vertex CP_{jm} to vertex CP_{ik} if $j = i$ and $k = m + 1$, or $j \neq i$ and there exists a message sent by processor p_j between CP_{jm} and $CP_{j(m+1)}$ and received by processor p_i between $CP_{i(k-1)}$ and CP_{ik}. Fig. 2 gives an example of CPG with its corresponding communication pattern.

Most of the ideas in this paper will be illustrated by the better visualized CPG instead of the more abstract poset. An element a in a poset is *maximal (minimal)* if there does not exist any element b such that $a < b$ ($b < a$); correspondingly, a vertex in a CPG will be referred as maximal (minimal) if it has no outgoing (incoming) edge. Also, the following

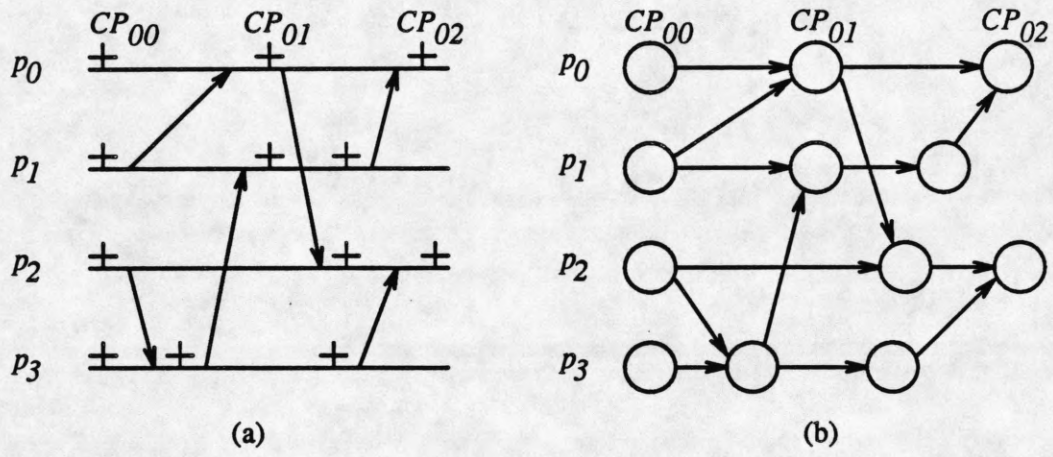


Figure 2: (a) The checkpoint and communication pattern; (b) the corresponding checkpoint graph

terminologies will be used interchangeably: $a < b$, a is "smaller than" b, b is "greater than" a, a can "strictly reach" b and b is "strictly reachable from" a.

B Maximum-Sized Antichain and Recovery Line

A partial ordering of a set S is *linear* if for every two elements a and b in S, either $a < b$ or $b < a$ [15]. In a poset, a subset whose elements are linearly ordered is called a *chain* and a set of elements, no two of which are comparable, is called an *antichain.* In particular, a set of any number of maximal (minimal) elements clearly forms an antichain. The antichains with the largest number of elements are called the *maximum-sized antichains* or *M-chains* for short. Let $\mathcal{A}(Q)$ denote the set of antichains on a poset Q and, for $A, B \in \mathcal{A}(Q)$, define $A \preceq B$ if and only if for all $a \in A$ there exists $b \in B$ such that $a \leq b$. [16]. Also let $\mathcal{M}(Q)$ denote the set of maximum-sized antichains. We then have the following properties.

LEMMA 1 *(1) $(\mathcal{A}(Q), \preceq)$ forms a poset;*

(2) $(\mathcal{A}(Q), \preceq)$ is a lattice and its subposet $(\mathcal{M}(Q), \preceq)$ is a sublattice;

(3) For $M_1, M_2 \in \mathcal{M}(Q)$, the join *(least upper bound) $M_1 \vee M_2 = max(M_1 \cup M_2)$ and the*

meet *(greatest lower bound)* $M_1 \wedge M_2 = min(M_1 \cup M_2)$, *where* $max(S)$ *denote the set of maximal elements in* S *and* $min(S)$ *is similarly defined [17, 16].*

Since $(\mathcal{M}(Q), \preceq)$ is a finite lattice, there must exist a unique maximum member $M^*(Q)$, called the *maximum maximum-sized antichain* or *MM-chain*, such that $M \preceq M^*(Q)$ for every $M \in \mathcal{M}(Q)$.

LEMMA 2 *For any* $M \in \mathcal{M}(Q)$, *there must not exist any* $a \in M^*(Q)$ *such that* $a < b$ *for* $b \in M$.

Proof. Suppose there exist such $a \in M^*(Q)$ and $b \in M$. $M \preceq M^*(Q)$ implies there exists $c \in M^*(Q)$ such that $b \leq c$. Together with $a < b$, this leads to $a < c$, contradicting the fact that $M^*(Q)$ is an antichain. □

In this paper, we define a *global checkpoint* to be a set of checkpoints, one from each processor. Based on the discussion on consistency in the previous section, a *consistent global checkpoint* is a set of checkpoints, one from each processor and no two of which are comparable through the "happens before" relation. A *recovery line* refers to the latest consistent global checkpoint. Because one special feature of the poset $P = (C, <)$ is that there always exists a natural *chain decomposition* $C_0, C_1, ..., C_{N-1}$ where C_i is the set of all checkpoints of processor p_i, the size $d(P)$ of the M-chains cannot be greater than N. Furthermore, because the first checkpoint of every processor must be minimal and the set of such checkpoints always forms an antichain of size N, $d(P)$ is equal to N and each M-chain will consist of N elements, one from each C_i. It becomes clear that each M-chain is equivalent to a consistent global checkpoint. Since it is always desirable to rollback to the most recent consistent global checkpoint in order to minimize the recovery cost, Lemma 1 guarantees the existence and uniqueness of such a recovery line, i.e., the MM-chain.

C Ideal, Filter and The Reachable Set

Given a poset P, if $\mathcal{I}$ is a set of elements of P with the property

$$a \in \mathcal{I} \text{ and } b \leq a \Longrightarrow b \in \mathcal{I},$$

$\mathcal{I}$ is called an *ideal* or a *down-set* of P. Similarly, a *filter* or an *up-set*, $\mathcal{F}$, of P is a set of elements such that if $a \in \mathcal{F}$ and $a \leq b$, then $b \in \mathcal{F}$.

For an antichain A in P, define

$$\mathcal{I}(A) = \{x \in P : x \leq a \text{ for some } a \in A\}$$

$$\mathcal{F}(A) = \{x \in P : x \leq a \text{ for some } a \in A\}.$$

Then $\mathcal{I}(A)$ is an ideal [16] and $\mathcal{F}(A)$ is a filter.

LEMMA 3 *A and B are antichains, then [16]*

(1)

$$\mathcal{I}(A) \subseteq \mathcal{I}(B) \Longleftrightarrow A \preceq B;$$

(2)

$$\mathcal{F}(A) \subseteq \mathcal{F}(B) \Longleftrightarrow B \preceq A.$$

In terms of the CPG, the set of vertices which can reach any vertex in an antichain A is equal to $\mathcal{I}(A)$ and the set of vertices reachable from any vertex in A is equal to $\mathcal{F}(A)$.

D The Rollback Propagation Algorithm

The algorithm for finding the recovery line will form the basis of our checkpoint space reclamation algorithm. The problem of finding the MM-chain in a general poset can be transformed into that of finding a maximum matching on a bipartite graph [18]. For the

poset $P = (C, <)$ in our problem, a simpler *rollback propagation algorithm*, shown in Fig. 3, has been proposed [4] and applied to the CPG.

```
/* CP stands for checkpoint */
/* Initially, all the CPs are unmarked */
include the latest CP of each processor in the root set;
mark all CPs strictly reachable from any CP in the root set;
while (at least one CP in the root set is marked) {
    replace each marked CP in the root set by the latest unmarked CP on the
    same processor;
    mark all CPs strictly reachable from any CP in the root set;
}
the root set is the recovery line.
```

Figure 3: The Rollback Propagation Algorithm

The complexity of the algorithm is linear in the number of edges because each edge can be removed after it is used to reach some vertex and therefore visited at most once.

IV PROBLEM FORMULATION

We first define the *potential recovery line* of a given checkpoint graph G as the recovery line of any checkpoint graph G' which can possibly evolve from G during program execution in the future. Since the purpose of keeping checkpoints is for possible future recovery, a checkpoint is *discardable* if and only if it does not belong to any potential recovery line. Being older than the current global recovery line is simply a sufficient condition for being a discardable checkpoint but not a necessary condition. We will show there exist checkpoints not older than the global recovery line yet discardable.

Scheme A and B for rollback recovery, as described in Section II, present different levels of difficulty for the problem of identifying discardable checkpoints. In Scheme B, a recovery line exists after each recovery such that all the older checkpoints can be discarded and all the newer checkpoints are invalidated by the rollback (Fig. 4(a) and (c)). Therefore, each recovery will start a new checkpoint graph and we only have to consider the potential recovery lines of the existing graph up to next recovery. Scheme A can be viewed as a more general case of Scheme B. Some checkpoints will be invalidated due to the rollback and result in a checkpoint graph which is a subgraph of the one before recovery (Fig. 4(a) and (b)). A checkpoint is discardable only when it will never belong to any recovery line no matter how many times the recovery occurs. We first consider Scheme B in the following sections and then show in Section VIII that the same necessary and sufficient conditions are also applicable to Scheme A.

Although the execution time for a normal program is finite, the possibility of augmenting the existing CPG by adding new vertices is enormous because the communication pattern is in general unpredictable. By recognizing the following rules for adding new vertices to a checkpoint graph, we are able to reduce the almost infinite number of situations to finite cases for the problem of identifying minimum number of non-discardable checkpoints. For each new vertex CP_{ik},

Rule 1: CP_{ik} must have an incoming edge from $CP_{i(k-1)}$ except for the first vertex on each chain which has no incoming edge;

Rule 2: CP_{ik} can have incoming edges from arbitrary existing vertices. But it can not have any outgoing edge to any existing vertex.

Note that a checkpoint CP_{ik} that happens before CP_{jm} may not be collected before CP_{jm}. However, such a situation can be detected by the communication information. If

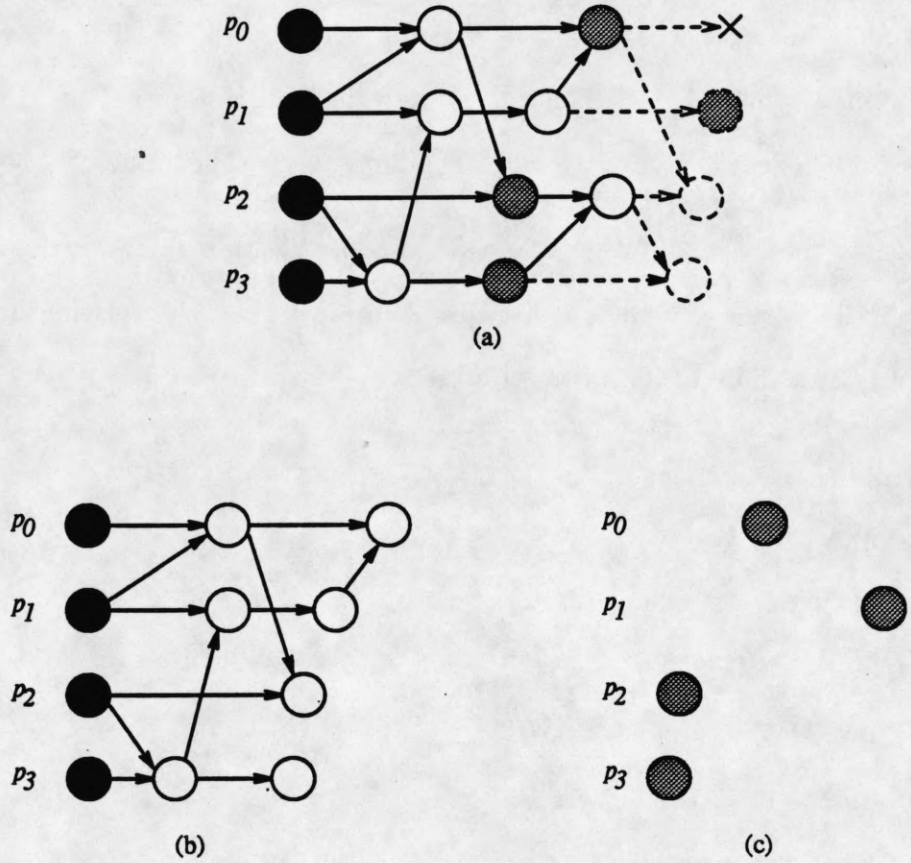


Figure 4: Checkpoint graphs (a) at the time of recovery; (b) after recovery with Scheme A; (c) after recovery with Scheme B. (Dashed lines represent the communication information from virtual checkpoints. Black vertices indicate the current global recovery line before recovery and gray vertices form the local recovery line at the time of recovery.)

a vertex CP_{jm} is expecting an incoming edge from a non-existing vertex CP_{ik}, CP_{jm} and its associated incoming edges will be excluded from the existing CPG. By adding each new vertex under this constraint, none of the new vertices can have edges pointing to any existing vertex and, therefore, Rule 2 is enforced. The following important property is ensured by Rule 2.

PROPERTY 1 *Adding a new vertex v and its associated incoming edges to an existing CPG can not change the relation between any pair of existing vertices.*

Proof. The relation between any pair of existing vertices will be changed only if one vertex is smaller than v and the other one is greater than v. However, Rule 2 guarantees none of the existing vertices is greater than v. Therefore, the property holds. □

Let $\mathcal{G}_f(G)$ denote the set of all future graphs obtainable by adding new vertices to a given checkpoint graph G according to the above rules. Lemma 4 gives the relationship between the antichains of G and those of its future graphs.

LEMMA 4 *Given $G = (V, E)$ and $G' \in \mathcal{G}_f(G)$,*

(1) $\mathcal{A}(G) \subseteq \mathcal{A}(G')$;

(2) $A \in \mathcal{A}(G')$ and $A \subseteq V \Longrightarrow A \in \mathcal{A}(G)$;

(3) $\mathcal{M}(G) \subseteq \mathcal{M}(G')$;

(4) $M \in \mathcal{M}(G')$ and $M \subseteq V \Longrightarrow M \in \mathcal{M}(G)$;

(5) $M^*(G) \preceq M^*(G')$.

Proof. (1) and (2) follow immediately from Property 1. By Rule 1 and the discussion after Lemma 2, the size of the maximum-sized antichains is always fixed and equal to the number of processors. So (3) and (4) holds. In particular, $M^*(G) \in \mathcal{M}(G) \subseteq \mathcal{M}(G')$ implies $M^*(G) \preceq M^*(G')$. □

Let $N_D(G)$ denote the set of non-discardable checkpoints of a given graph G. By definition, we have

$$N_D(G) = \{v : v \in G \text{ and } v \in M^*(G') \text{ for some } G' \in \mathcal{G}_f(G)\}. \quad (1)$$

Our goal is to develop an algorithm to efficiently find the set $N_D(G)$.

V THE SET OF NON-DISCARDABLE CHECKPOINTS

One feature of Scheme B is that the checkpoint graph is always growing until a rollback recovery occurs, after which a new checkpoint graph starts. In this case, it is clear by the definition in Eq. 1 that $N_D(G') \cap G \subseteq N_D(G)$ for any $G' \in \mathcal{G}_f(G)$ because $\mathcal{G}_f(G') \subseteq \mathcal{G}_f(G)$. In other words, once a checkpoint is determined to be discardable, it will never become non-discardable for any future graph. Note that a discardable checkpoint v can be removed from the graph by the following procedure without affecting any recovery line in the future.

1. A new edge is generated for each pair of incoming and outgoing edges of v in order to preserve the relations implied through v among the remaining vertices.

2. The source vertices of all the incoming edges of v have to be remembered. When an outgoing edge of v is added in the future, it is replaced by the outgoing edges from these source vertices.

However, since we are mainly concerned about checkpoint space reclamation, we will leave all the vertices corresponding to discardable checkpoints in the graph for simplicity.

One special future graph of G, $\hat{G}$, will play a very important role throughout this paper and is constructed as follows:

1. adjoin N new vertices $n_0, n_1, \ldots, n_{N-1}$ to G;

2. an edge is added from the last vertex e_i on each chain C_i to n_i as shown in Fig. 5.

Let V_n denote the set of all such n_i's and V_e denote the set of e_i's. We now prove the following necessary and sufficient conditions for a checkpoint to be non-discardable.

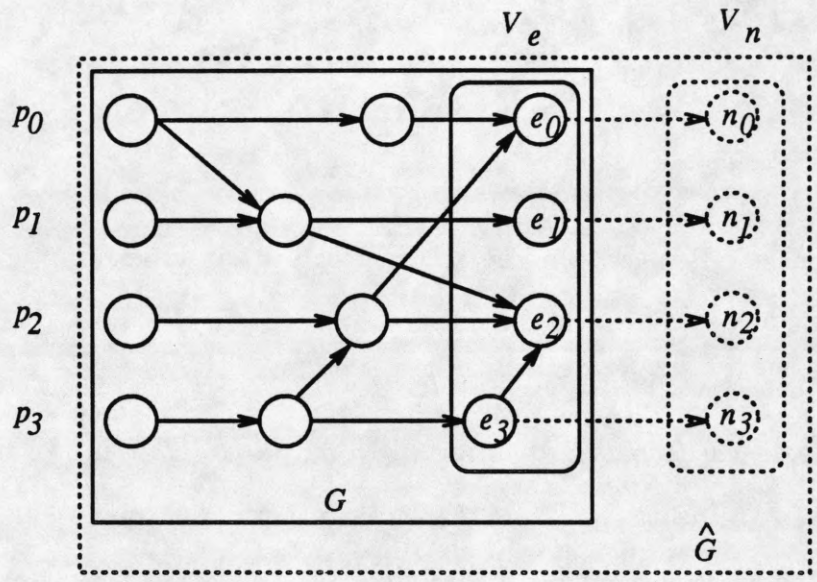


Figure 5: Construction of the future graph $\hat{G}$ by adding n_i's to the checkpoint graph G.

THEOREM 1 *Given a checkpoint graph $G = (V, E)$ and $v \in G$,*

$$v \in M^*(G') \text{ for some } G' = (V', E') \in \mathcal{G}_f(G)$$

$$\text{if and only if} \quad v \in M^*(\hat{G} - W) \text{ for some } W \subseteq V_n.$$

Proof. The *if* part is trivial because $\hat{G} - W \in \mathcal{G}_f(G)$. We now prove the *only if* part. If $v \in M^*(G')$ *for some* $G' \in \mathcal{G}_f(G)$, let $M^*(G') = M_1 \cup M_2$ such that $M_1 = M^*(G') \cap V$ and $M_2 = M^*(G') \setminus M_1$ as shown in Fig. 6(a). In particular, $v \in M_1$. Define $p(u) = i$ if u represents a checkpoint of processor p_i and decompose the set V_n as $V_n = B_1 \cup B_2$ where

$$B_1 = \{n_{p(u)} : u \in M_1\}$$

$$B_2 = \{n_{p(u)} : u \in M_2\}.$$

We want to show that $M_1 \cup B_2 = M^*(\hat{G} - B_1)$ (Fig. 6(b)).

First we prove $M_1 \cup B_2 \in \mathcal{M}(\hat{G} - B_1)$. Consider the graph G'. For every $u \in M_2$, $e_{p(u)} < u$ by Rule 1. According to Lemma 3, $\mathcal{I}(e_{p(u)}) \subseteq \mathcal{I}(u)$. Since u and all the vertices in

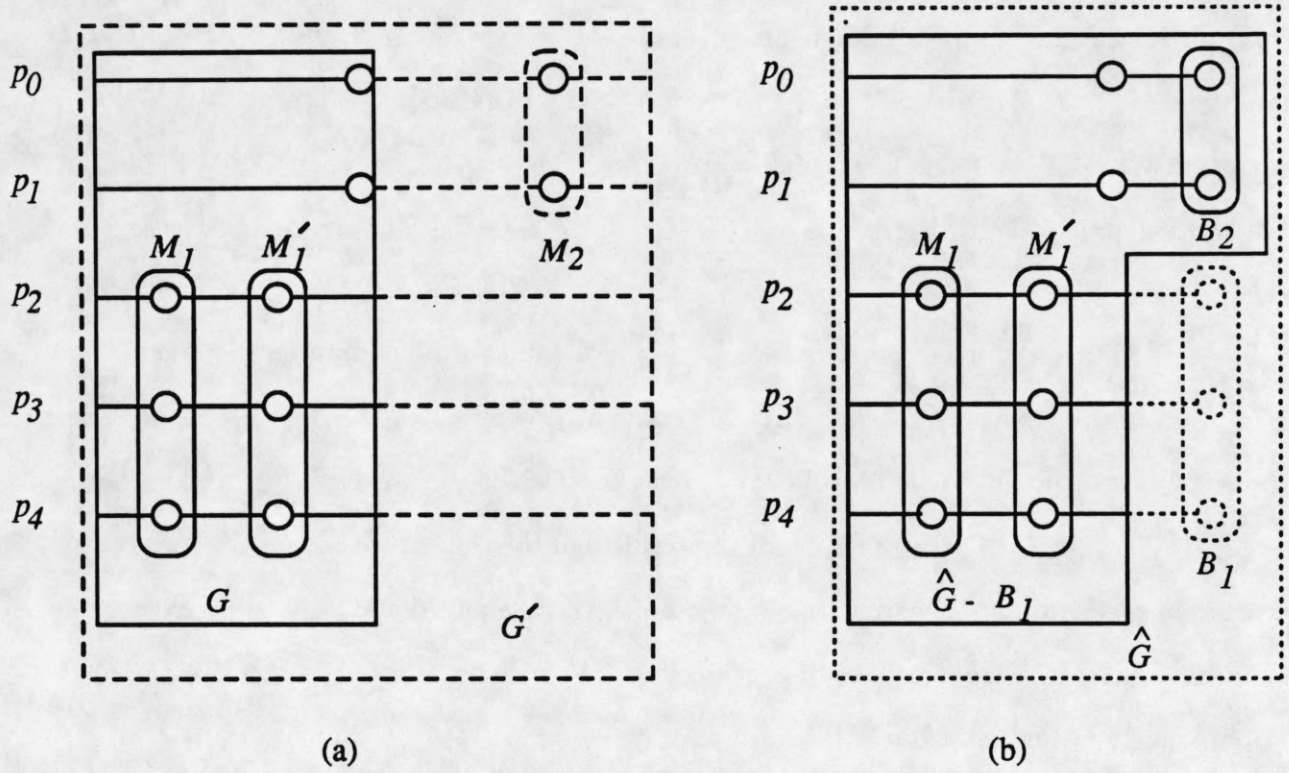


Figure 6: Suppose (a) $M_1 \cup M_2$ forms the MM-chain of G', then (b) $M_1 \cup B_2$ forms the MM-chain of $\hat{G} - B_1$.

M_1 belong to the same antichain, $M_1 \cap \mathcal{I}(u) = \emptyset$. It follows that $M_1 \cap \mathcal{I}(e_{p(u)}) = \emptyset$. Now consider the graph $\hat{G} - B_1$. The above equation still holds because of Property 1. By the construction of $\hat{G}$, $\mathcal{I}(n_{p(u)}) = \mathcal{I}(e_{p(u)}) \cup \{n_{p(u)}\}$ and therefore $M_1 \cap \mathcal{I}(n_{p(u)}) = \emptyset$. Since Rule 2 implies $M_1 \cap \mathcal{F}(n_{p(u)}) = \emptyset$, we have proved that every vertex $n_{p(u)}$ in B_2 is incomparable with every vertex in M_1. Because M_1 is an antichain by itself and all $n_{p(u)}$'s in B_2 are maximal, $M_1 \cup B_2 \in \mathcal{M}(\hat{G} - B_1)$.

Next we prove $M_1 \cup B_2 = M^*(\hat{G} - B_1)$ by contradiction. Because every vertex in B_2 is maximal on the chain it belongs to, $B_2 \subseteq M^*(\hat{G} - B_1)$ by Lemma 2. Suppose $M_1 \cup B_2 \neq M^*(\hat{G} - B_1)$. There must exist $M_1' = M^*(\hat{G} - B_1) \setminus B_2$ such that $M_1 \preceq M_1'$ and $M_1 \neq M_1'$. We then have $\mathcal{F}(M_1') \subseteq \mathcal{F}(M_1)$ by Lemma 3. Recall M_1 and M_2 form an antichain in the

graph G', which implies $M_2 \cap \mathcal{F}(M_1) = \emptyset$. Thus $M_2 \cap \mathcal{F}(M_1') = \emptyset$ and, because Rule 2 guarantees $M_2 \cap \mathcal{I}(M_1') = \emptyset$, $M_1' \cup M_2 \in \mathcal{M}(G')$. The fact that $M_1' \cup M_2$ is a greater M-chain than $M_1 \cup M_2$ in G' contradicts $M^*(G') = M_1 \cup M_2$. Hence, $M_1 \cup B_2 = M^*(\hat{G} - B_1)$.

It immediately follows that if $v \in M^*(G')$ *for some* $G' \in \mathcal{G}_f(G)$, $v \in M_1 \subseteq M^*(\hat{G} - B_1)$ for $B_1 \subseteq V_n$. □

The contribution of Theorem 1 is that it classifies the enormous number of possibilities for any existing vertex to become a member of some MM-chain in the future into exponential number of cases. If we apply the rollback propagation algorithm to each of the 2^N graphs $\hat{G} - W$, $W \subseteq V_n$, and take the union of all the resulting MM-chains, we obtain the set of non-discardable checkpoints. However, this is an exponential algorithm and may be unacceptable for applications with large number of processors. We will give another theorem in the next section which can further reduce the number of cases.

VI THE MAXIMUM CHECKPOINT SPACE RECLAMATION ALGORITHM

By applying Lemma 1, we will show that each of the 2^N MM-chains in Theorem 1 can be "synthesized" by N MM-chains. An efficient algorithm is then developed for finding the set of non-discardable checkpoints.

LEMMA 5 *Given a poset* $P = (S, \preceq)$ *and* $A, B \subseteq S$,

$$min(A \cup B) = min(min(A) \cup B).$$

Proof. Let $min'(A) = A \setminus min(A)$. By definition, for each $a' \in min'(A)$, there exists $a \in A$ such that $a \preceq a'$ and $a \neq a'$. Since $a,\ a' \in A \cup B$, $a' \notin min(A \cup B)$. Therefore,

$$min(A \cup B) = min(min(A) \cup min'(A) \cup B) = min(min(A) \cup B).$$ □

LEMMA 6 *Given a poset P, $M \in \mathcal{M}(P)$ and $M \preceq M_i \in \mathcal{M}(P)$ for $i \in [0, k-1]$ for any finite k. Define*

$$\bigwedge_{i\in[0,k-1]} M_i = (...((M_0 \wedge M_1) \wedge M_2) \ ... \ \wedge M_{k-1},$$

then

(1)

$$M \preceq \bigwedge_{i\in[0,k-1]} M_i \ \in \mathcal{M}(P);$$

(2)

$$\bigwedge_{i\in[0,k-1]} M_i = min(\bigcup_{i\in[0,k-1]} M_i).$$

Proof. Both parts will be proved by induction on k.

(1) By Lemma 1, $\mathcal{M}(P)$ is a lattice and so $M_0 \wedge M_1 \in \mathcal{M}(P)$. Also, $M \preceq M_0 \wedge M_1$ because $M \preceq M_0$, $M \preceq M_1$ and $M_0 \wedge M_1$ is the greatest lower bound of M_0 and M_1. We have shown the case $k = 2$ is true. Assume it is true for $k = n - 1$, i.e.

$$M \preceq \bigwedge_{i\in[0,n-2]} M_i \ \in \mathcal{M}(P). \tag{2}$$

Again, by Lemma 1,

$$\bigwedge_{i\in[0,n-1]} M_i = (\bigwedge_{i\in[0,n-2]} M_i) \wedge M_{n-1} \in \mathcal{M}(P).$$

Eq. 2 and $M \preceq M_{n-1}$ implies

$$M \preceq \bigwedge_{i\in[0,n-1]} M_i.$$

Therefore, it is also true for $k = n$ and so we have (1).

(2) The case $k = 2$ is true by Lemma 1. Assume it is true for $k = n - 1$, i.e.

$$\bigwedge_{i\in[0,n-2]} M_i = min(\bigcup_{i\in[0,n-2]} M_i). \tag{3}$$

Applying part (1), Eq. 3 and Lemma 1, we have

$$\bigwedge_{i\in[0,n-1]} M_i = (\bigwedge_{i\in[0,n-2]} M_i) \wedge M_{n-1} = min(min(\bigcup_{i\in[0,n-2]} M_i) \cup M_{n-1}).$$

Lemma 5 further gives that

$$min(min(\bigcup_{i\in[0,n-2]} M_i) \cup M_{n-1}) = min(\bigcup_{i\in[0,n-2]} M_i \cup M_{n-1}) = min(\bigcup_{i\in[0,n-1]} M_i).$$

Therefore, by induction, part (2) is true. □

THEOREM 2 *For every $W \subseteq V_n$,*

$$M^*(\hat{G} - W) = min(\bigcup_{n_i\in W} M^*(\hat{G} - n_i)).$$

Proof. If there are k vertices in the set W, without loss of generality, we may assume they are $z_0, z_1, ..., z_{k-1}$, i.e.

$$\{n_i : n_i \in W\} = \{z_j : j \in [0, k-1]\}.$$

Since $\hat{G} - z_j \in \mathcal{G}_f(\hat{G} - W)$, $M^*(\hat{G} - W) \preceq M^*(\hat{G} - z_j)$ for all $j \in [0, k-1]$ by Lemma 4.

Now consider the graph $\hat{G}$. $\hat{G} \in \mathcal{G}_f(\hat{G} - z_j)$ implies that $M^*(\hat{G} - z_j) \in \mathcal{M}(\hat{G})$ for $j \in [0, k-1]$. Similarly, $M^*(\hat{G} - W) \in \mathcal{M}(\hat{G})$. By Lemma 6,

$$M^*(\hat{G} - W) \preceq \bigwedge_{j\in[0,k-1]} M^*(\hat{G} - z_j) = min(\bigcup_{j\in[0,k-1]} M^*(\hat{G} - z_j)) \in \mathcal{M}(\hat{G}). \quad (4)$$

Moreover, for every $j \in [0, k-1]$, there exists $u \in M^*(\hat{G} - z_j)$ with $p(u) = p(z_j)$ such that $u < z_j$. Since $u \in \bigcup_{j\in[0,k-1]} M^*(\hat{G} - z_j)$, $z_j \notin min(\bigcup_{j\in[0,k-1]} M^*(\hat{G} - z_j))$. We have, by Lemma 4, $min(\bigcup_{j\in[0,k-1]} M^*(\hat{G} - z_j)) \in \mathcal{M}(\hat{G} - W)$ and

$$min(\bigcup_{j\in[0,k-1]} M^*(\hat{G} - z_j)) \preceq M^*(\hat{G} - W).$$

Together with Eq. 4, we have proved

$$M^*(\hat{G} - W) = min(\bigcup_{j\in[0,k-1]} M^*(\hat{G} - z_j)) = min(\bigcup_{n_i\in W} M^*(\hat{G} - n_i)).$$ □

In particular, the current global recovery line, $M^*(G)$, can be obtained by letting $W = V_n$,

$$M^*(G) = min(\bigcup_{i\in[0,N-1]} M^*(\hat{G} - n_i)).$$

COROLLARY 1 *Given a checkpoint graph $G = (V, E)$,*

$$N_D(G) = \bigcup_{i\in[0,N-1]} M^*(\hat{G} - n_i) \cap V$$

Proof. For any $v \in \bigcup_{i\in[0,N-1]} M^*(\hat{G} - n_i) \cap V$, $v \in M^*(\hat{G} - n_i)$ for some $i \in [0, N-1]$. Since $\hat{G} - n_i \in \mathcal{G}_f(G)$, $v \in N_D(G)$ by definition. Thus $\bigcup_{i\in[0,N-1]} M^*(\hat{G} - n_i) \cap V \subseteq N_D(G)$.

Conversely, for any $v \in N_D(G)$, $v \in V$ and $v \in M^*(\hat{G} - W)$ for some $W \subseteq V_n$ by Theorem 1. Theorem 2 further gives that

$$v \in min(\bigcup_{n_i\in W} M^*(\hat{G} - n_i)) \subseteq \bigcup_{n_i\in W} M^*(\hat{G} - n_i) \subseteq \bigcup_{i\in[0,N-1]} M^*(\hat{G} - n_i).$$

Therefore, $N_D(G) \subseteq \bigcup_{i\in[0,N-1]} M^*(\hat{G} - n_i) \cap V$ and so we have

$$N_D(G) = \bigcup_{i\in[0,N-1]} M^*(\hat{G} - n_i) \cap V. \qquad \square$$

The above corollary provides the basis of a good algorithm for finding $N_D(G)$. We now present the Maximum Checkpoint Space Reclamation (MCSR) algorithm in Fig. 7. The algorithm is of complexity $\mathbf{O}(N|E|)$, where $|E|$ is the total number of edges in the checkpoint graph.

Fig 8 shows the checkpoint graph corresponding to the initial part of execution of an N-Queen program written in Chare Kernel language [19] which has been developed as a medium-grain, message-driven and machine-independent parallel language at the University of Illinois. Some edges that do not affect the result of applying the MCSR algorithm are

/* N is the number of processors */
/* $\hat{G}$ and n_i are as defined in the beginning of Section V */
for each $i \in [0, N-1]$ {
apply the rollback propagation algorithm on the checkpoint graph $\hat{G} - n_i$ to find the recovery line;
add to the set $N_D(G)$ the checkpoints in G and on the recovery line;
}
all the checkpoints not in $N_D(G)$ can be reclaimed.

Figure 7: The Maximum Checkpoint Space Reclamation Algorithm

removed in order to get a clear picture. Denote the set of non-discardable vertices contributed by $M^*(\hat{G} - n_i)$ as N_{Di}, we have

$$N_{D0} = \{a\}, \quad N_{D1} = \{b\}, \quad N_{D2} = \{c\},$$

$$N_{D3} = N_{D4} = N_{D5} = \{d, e, f, g, h, i\}.$$

showed as the shaded vertices in Fig. 8. Traditional checkpoint space reclamation algorithms can not reclaim any of the checkpoints due to the domino effect. It is interesting that the MCSR algorithm determines that all non-shaded checkpoints can be reclaimed.

VII THE MAXIMUM NUMBER OF NON-DISCARDABLE CHECKPOINTS

Traditionally, the checkpoint space reclamation procedure is only performed for the set of checkpoints older than the current global recovery line. Since it is possible for the domino effect to persist during the program execution, a common perception is that all checkpoints

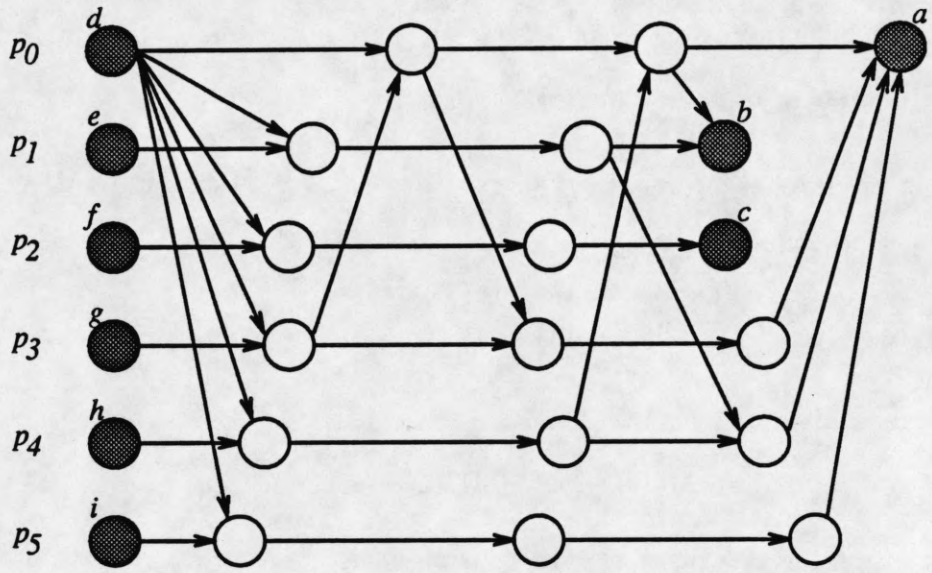


Figure 8: Checkpoint graph corresponding to part of the execution of an N-Queen program

may have to be kept and the space overhead may be constantly growing as a program proceeds. In a sense, this is a more serious disadvantage than the slower recovery due to the domino effect because it results in unpredictable space overhead during normal execution. Corollary 1 not only identifies the minimum set of non-discardable checkpoints but also places an upper bound N^2 on the number of non-discardable checkpoints for a general checkpoint graph because each $M^*(\hat{G} - n_i)$, $i \in [0, N-1]$, consists of N checkpoints. A smaller upper bound obviously exists based on the following observation:

1. $M^*(\hat{G} - n_i)$ may contain vertices from ths set V_n, but we are only concerned about vertices in the existing graph G;

2. $M^*(\hat{G} - n_i)$'s may not be mutually disjoint;

3. if the last vertex, e_i, on chain C_i is maximal, $M^*(\hat{G} - n_i)$ will contribute only a single vertex to the set $N_D(G)$, i.e. e_i itself.

The following lemma addresses the implicit relations among $M^*(\hat{G} - n_i)$'s.

LEMMA 7 *Let m_{ij} denote the vertex in $M^*(\hat{G}-n_i)$ from processor p_j. For $i,j \in [0,N-1]$ and $i \neq j$, if $m_{ij} \neq n_j$ and $m_{ji} \neq n_i$, then $M^*(\hat{G}-n_i) = M^*(\hat{G}-n_j)$.*

Proof. $m_{ij} \neq n_j$ implies $M^*(\hat{G}-n_i) \subseteq \hat{G}-n_i-n_j$. By Lemma 4, $M^*(\hat{G}-n_i) \in \mathcal{M}(\hat{G}-n_i-n_j) \subseteq \mathcal{M}(\hat{G}-n_j)$ and so

$$M^*(\hat{G}-n_i) \preceq M^*(\hat{G}-n_j).$$

Similarly, $m_{ji} \neq n_i$ leads to

$$M^*(\hat{G}-n_j) \preceq M^*(\hat{G}-n_i).$$

Since $\mathcal{M}(\hat{G}-n_i-n_j)$ forms a poset (Lemma 1), we have

$$M^*(\hat{G}-n_i) = M^*(\hat{G}-n_j). \qquad \square$$

THEOREM 3 *For any checkpoint graph $G=(V,E)$,*

$$|N_D(G)| \leq \frac{N(N+1)}{2}.$$

Proof. By Corollary 1, we only have to consider the N^2 vertices m_{ij}, $i,j \in [0,N-1]$. For each $i \in [0,N-1]$, $m_{ii} \in V$ and contributes one vertex to $N_D(G)$. Since all the m_{ii}'s come from different processors, N_D now consists of N vertices. For the remaining N^2-N vertices with $i \neq j$, we consider each pair m_{ij} and m_{ji} at a time and there are $(N^2-N)/2$ such pairs. We distinguish three cases:

Case 1: $m_{ij}=n_j$ and $m_{ji}=n_i$. Both m_{ij} and m_{ji} do not belong to $N_D(G)$.

Case 2: $m_{ij}=n_j$ and $m_{ji} \neq n_i$, or $m_{ij} \neq n_j$ and $m_{ji}=n_i$. This pair will possibly add one new vertex to $N_D(G)$.

Case 3: $m_{ij} \neq n_j$ and $m_{ji} \neq n_i$. It follows that $M^*(\hat{G}-n_i) = M^*(\hat{G}-n_j)$ by Lemma 7, and so $m_{ij}=m_{jj}$ and $m_{ji}=m_{ii}$. Since m_{jj} and m_{ii} are already in $N_D(G)$, this case does not increase the size of $N_D(G)$.

Therefore, each of the $(N^2-N)/2$ pairs can contribute at most one new vertex to $N_D(G)$. We then have

$$|N_D(G)| \leq N + \frac{N^2 - N}{2} = \frac{N(N+1)}{2}. \qquad \square$$

One may argue that the upper bound derived in Theorem 3 is still of the order N^2. We will next show that $N(N+1)/2$ is in fact the lowest upper bound, i.e., the maximum, because for any N we can construct a checkpoint graph, G_N^*, as shown in Fig. 9 to achieve this upper bound. By applying the MCSR algorithm in Fig. 7, it is not hard to see that all the $N(N+1)/2$ vertices in Fig. 9 are non-discardable.

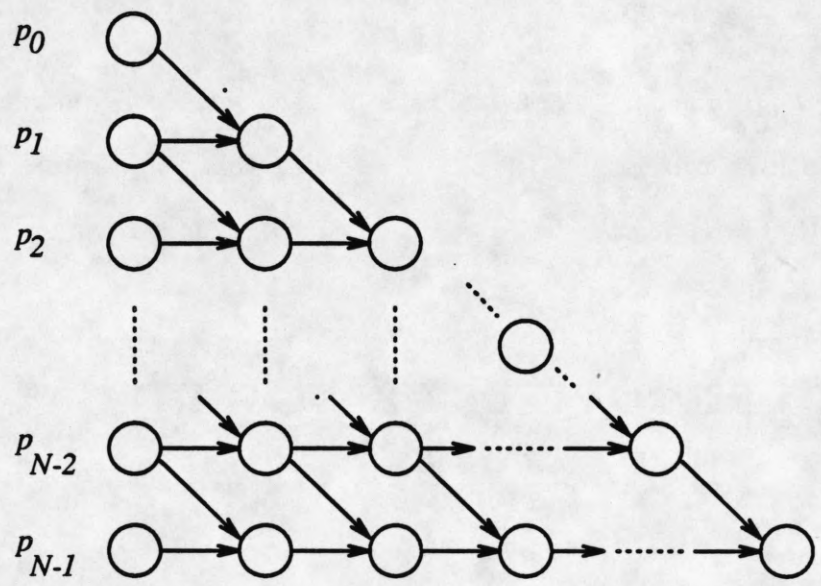


Figure 9: G_N^*: The checkpoint graph with $N(N+1)/2$ non-discardable checkpoints

When a checkpoint graph is given, we can further reduce the maximum by counting the number of maximal vertices, L, in the set V_e (as shown in Fig. 5). Recall that if e_i is maximal, $m_{ii} = e_i$ and $m_{ij} = n_j$ for $j \neq i$. Therefore, in the discussion for each pair of m_{ij} and m_{ji} in the proof of Theorem 3, the case when both e_i and e_j are maximal corresponds to Case 1.

The maximum then becomes

$$|N_D(G)| \leq N + \binom{N}{2} - \binom{L}{2}.$$

In particular when $L = N$, $|N_D(G)| = N$, which is corresponding to the case of coordinated checkpointing.

We would like to point out that the MCSR algorithm can be further improved by applying Lemma 7. Inside the loop in Fig. 7, suppose we have found the recovery line $M^*(\hat{G} - n_i)$. Define the index set Γ as

$$\Gamma = \{j : m_{ij} \neq n_j,\ j \in [0, N-1] \ and \ j > i\}.$$

Then for each later loop index $j \in \Gamma$, the rollback propagation algorithm can be aborted when any checkpoint from processor p_i is marked. Because this would mean $m_{ji} \neq n_i$ and $M^*(\hat{G} - n_j)$ is exactly the same as $M^*(\hat{G} - n_i)$.

VIII RECOVERY USING VIRTUAL CHECKPOINTS

In this section, we want to show that our results in previous sections can also be applied to Scheme A, i.e., recovery using virtual checkpoints. When a processor p_i detects an error and decides to rollback, all the computation after its latest checkpoint are assumed invalid. However, for each of the non-faulty processors, the state residing in the volatile storage is still valid and can serve as a new checkpoint to advance the recovery line. Since the checkpoints belonging to the current global recovery line will still remain on the stable storage, the virtual checkpoints can be discarded after the recovery and disappear from the checkpoint graph.

The effect of performing such recovery on the checkpoint graph is described as follows. The existing checkpoint graph $G = (V, E)$ is augmented by the set of virtual checkpoints at the time of recovery and becomes graph G'. $M^*(G')$ is then obtained by the rollback propagation algorithm. Processes roll back according to $M^*(G')$ and the vertices belonging to the set $\mathcal{F}(M^*(G')) \setminus M^*(G')$ are deleted from G', together with all the associated edges. Finally, all the vertices representing the virtual checkpoints are deleted.

Our approach to solving this more general problem when the checkpoint graph is not always growing is described as follows. First, we apply the results from previous sections to the checkpoint graphs before the first rollback. Since $G' \in \mathcal{G}_f(G)$, $M^*(G')$ will not contain any discardable checkpoints of G. Next, we again apply previous results to the checkpoint graphs after the first rollback and before the second rollback to determine the set of non-discardable checkpoints. If we can show that the rollback procedure does not make any discardable checkpoint become non-discardable, i.e., needed for any possible second rollback, the MCSR algorithm is then valid for Scheme A as well as for Scheme B because once a checkpoint is determined to be discardable it will remain discardable no matter how many times rollback recovery occurs in the future. Note that if the set of deleted vertices can be arbitrary, the above statement may not be true. For example, for the checkpoint graph G shown in Fig. 10(a), vertex CP_{12} is found to be discardable by the MCSR algorithm. However, in the graph $G - CP_{02}$ shown in Fig. 10(b), CP_{12} becomes non-discardable.

The proof of Theorem 1 characterizes, as a byproduct, the possible sets of deleted vertices due to rollback. That is, for any $G' \in \mathcal{G}_f(G)$, $M^*(G') \cap V \subseteq M^*(\hat{G} - T)$ where

$$T = \{n_{p(u)} : u \in M^*(G') \cap V\}. \tag{5}$$

Define a *strict filter* corresponding to such T as

$$\mathcal{F}_s(T) = \mathcal{F}(M^*(\hat{G} - T)) \setminus M^*(\hat{G} - T)$$

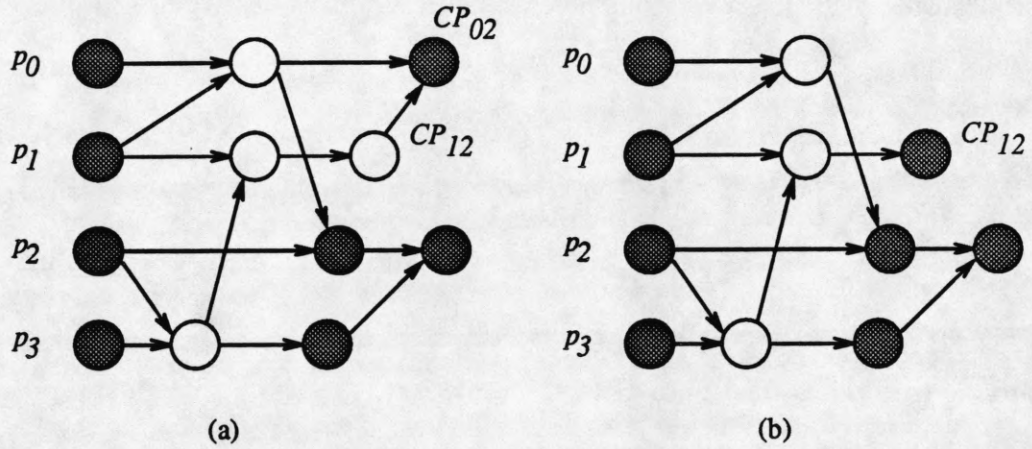


Figure 10: Discardable checkpoint CP_{12} becomes non-discardable after the removal of checkpoint CP_{02}. (Shaded vertices are non-discardable according to the MCSR algorithm.)

and denote

$$G(T) = G - \mathcal{F}_s(T), \tag{6}$$

it suffices to prove that $N_D(G(T)) \subseteq N_D(G)$. Parallel to the definitions of e_i, n_i, V_n and $\hat{G}$ for the graph G, we define e_i', n_i', V_n' and $\hat{G}(T)$ for the graph $G(T)$. Clearly, for each $i \in [0, N-1]$ and $n_i \notin T$, $e_i' = e_i$ and $n_i' = n_i$. Also define

$$T' = \{n_i' : n_i \in T\}.$$

LEMMA 8 *For every $W \subseteq V_n \setminus T$,*

$$\hat{G} - (T \cup W) \quad \in \quad \mathcal{G}_f(\hat{G}(T) - (T' \cup W)).$$

Proof. Clearly, by definition,

$$(\hat{G} - T) - \mathcal{F}_s(T) = \hat{G}(T) - T'.$$

Since $V_n \setminus T \subseteq V_n'$, $W \subseteq V_n'$ and so $(\hat{G} - (T \cup W)) - \mathcal{F}_s(T) = \hat{G}(T) - (T' \cup W)$. Now we have to show all the vertices in $\mathcal{F}_s(T)$ and their associated edges can be added to $\hat{G}(T) - (T' \cup W)$,

following Rule 1 and 2, to obtain $\hat{G}-(T\cup W)$. Rule 1 is obviously satisfied. By always adding the smaller vertices first, Rule 2 is enforced among the vertices in $\mathcal{F}_s(T)$ during the processor. Suppose Rule 2 is violated when $v \in \mathcal{F}_s(T)$ is added, i.e. there exists $u \in \hat{G}(T) - (T' \cup W)$ such that an edge is drawn from v to u. Clearly, $u \notin V'_n \setminus (T' \cup W)$. If $u \in G(T)$, $v < u$ and $v \in \mathcal{F}_s(T)$ implies $u \in \mathcal{F}_s(T)$ by the definition of $\mathcal{F}_s(T)$. This contradicts the fact that $G(T) \cap \mathcal{F}_s(T) = \emptyset$. Therefore, Rule 2 is also satisfied. □

THEOREM 4 *Given a checkpoint graph $G = (V, E)$, for every $G' \in \mathcal{G}_f(G)$,*

$$N_D(G(T)) \subseteq N_D(G)$$

where T and $G(T)$ are as defined in Eqs. 5 and 6.

Proof. If $v \in N_D(G(T))$, $v \in M^*(G'(T))$ for some $G'(T) \in \mathcal{G}_f(G(T))$. Denote $G(T) = (V(T), E(T))$ and $G'(T) = (V'(T), E'(T))$. Let $M^*(G'(T)) = M_1 \cup M_2$ where $M_1 = M^*(G'(T)) \cap V(T)$ and $M_2 = M^*(G'(T)) \setminus M_1$. Clearly, $v \in M_1$. Also define

$$M_3 = \{e'_{p(u)} : u \in M_2 \text{ and } n_{p(u)} \in T\}.$$

Decompose V_n as $V_n = B_1 \cup B_2 \cup T$ where $B_1 = \{n_{p(u)} : u \in M_1\} \setminus T$ and $B_2 = \{n_{p(u)} : u \in M_2\} \setminus T$. Fig. 11 illustrates the above notation. We want to prove that

$$M_1 \cup M_3 \cup B_2 = M^*(\hat{G} - (T \cup B_1)).$$

First we show $M_1 \cup M_3 \cup B_2 \in \mathcal{M}(\hat{G} - (T \cup B_1))$. Recall that $M_3 \cup B_2 \subseteq M^*(\hat{G} - T)$ and so $M_3 \cup B_2 \in \mathcal{A}(\hat{G} - T)$. Since $\hat{G} - T \in \mathcal{G}_f(\hat{G}(T) - T')$ by the above lemma and $(M_3 \cup B_2) \cap \mathcal{F}_s(T) = \emptyset$, we have $M_3 \cup B_2 \in \mathcal{A}(\hat{G}(T) - T')$ by Lemma 4. By the same argument, $M_3 \cup B_2 \in \mathcal{A}(\hat{G}(T) - (T' \cup B_1))$. Following the proof of Theorem 1, we have $M_1 \cap \mathcal{I}(w) = \emptyset$ for all $w \in M_3 \cup B_2$. Since all the vertices in $M_3 \cup B_2$ are maximal in $\hat{G}(T) - (T' \cup B_1)$

(Fig. 11(b)), $M_1 \cap \mathcal{F}(M_3 \cup B_2) = \emptyset$ and so $M_1 \cup M_3 \cup B_2 \in \mathcal{M}(\hat{G}(T) - (T' \cup B_1))$. It follows that $M_1 \cup M_3 \cup B_2 \in \mathcal{M}(\hat{G} - (T \cup B_1))$ again by applying Lemmas 8 and 4.

Now suppose $M_1 \cup M_3 \cup B_2 \neq M^*(\hat{G} - (T \cup B_1))$. Since $\hat{G} - T \in \mathcal{G}_f(\hat{G} - (T \cup B_1))$, Lemma 4 gives $M^*(\hat{G} - (T \cup B_1)) \preceq M^*(\hat{G} - T)$. By Lemma 2, the facts that $M_3 \cup B_2 \subseteq M^*(\hat{G} - T)$ and $M_1 \cup M_3 \cup B_2 \in \mathcal{M}(\hat{G} - (T \cup B_1))$ implies $M_3 \cup B_2 \subseteq M^*(\hat{G} - (T \cup B_1))$. Therefore, there must exists (Fig. 11(c))

$$M_1' = M^*(\hat{G} - (T \cup B_1)) \setminus (M_3 \cup B_2)$$

such that $M_1 \preceq M_1'$ and $M_1 \neq M_1'$. We now have $\mathcal{F}(M_1') \subseteq \mathcal{F}(M_1)$. Together with $M_2 \cap \mathcal{F}(M_1) = \emptyset$, we get $M_2 \cap \mathcal{F}(M_1') = \emptyset$ and so $M_1' \cup M_2 \in \mathcal{M}(G'(T))$. This is a contradiction to $M^*(G'(T)) = M_1 \cup M_2$ because $M_1 \cup M_2 \preceq M_1' \cup M_2$ and $M_1 \cup M_2 \neq M_1' \cup M_2$. Therefore, we must have $M_1 \cup M_3 \cup B_2 = M^*(\hat{G} - (T \cup B_1))$.

Finally,

$$v \in M_1 \subseteq M^*(\hat{G} - (T \cup B_1)) \subseteq N_D(G)$$

and so $N_D(G(T)) \subseteq N_D(G)$. □

IX CONCLUSIONS

The problem of finding recovery lines for message-passing systems using independent checkpointing is formulated as determining the maximum maximum-sized antichains of partially ordered sets. We present a method for predicting the possibility for a checkpoint to become a member of some recovery line in the future, and show that some of the checkpoints will never be needed for recovery so their space can be reclaimed. Based on the algorithm for finding the recovery lines, a maximum checkpoint space reclamation algorithm,

with complexity linear in the number of processors N and linear in the number of edges in the checkpoint graph, is developed for determining the set of non-discardable checkpoints. The maximum, $N(N+1)/2$, of the number of non-discardable checkpoints for an arbitrary checkpoint graph is also derived to show that the space overhead for maintaining multiple checkpoints is bounded even when the domino effect persists during program execution.

ACKNOWLEDGEMENT

The authors wish to express their sincere thanks to Douglas West, Weiping Shi, Shyh-Kwei Chen and Sanjeev Khanna for their valuable discussions, to Balkrishna Ramkumar for his help with the Chare Kernel, and to L. V. Kalé for access to the Kernel.

REFERENCES

[1] Y. Tamir and C. H. Sequin, "Error recovery in multicomputers using global checkpoints," in *Proc. Int. Conf. on Parallel Processing*, pp. 32–41, 1984.

[2] K. M. Chandy and L. Lamport, "Distributed snapshots: Determining global states of distributed systems," *ACM Trans. on Computer Systems*, vol. 3, pp. 63–75, Feb. 1985.

[3] R. Koo and S. Toueg, "Checkpointing and rollback-recovery for distributed systems," *IEEE Trans. on Software Engineering*, vol. SE-13, pp. 23–31, Jan. 1987.

[4] K. Tsuruoka, A. Kaneko, and Y. Nishihara, "Dynamic recovery schemes for distributed processes," in *Proc. IEEE 2nd Symp. on Reliability in Distributed Software and Database*, pp. 124–130, 1981.

[5] R. E. Strom and S. Yemini, "Optimistic recovery in distributed systems," *ACM Trans. on Computer Systems*, vol. 3, pp. 204–226, Aug. 1985.

[6] B. Bhargava and S. R. Lian, "Independent checkpointing and concurrent rollback for recovery - An optimistic approach," in *Proc. IEEE Symp. on Reliable Distributed Systems*, pp. 3–12, 1988.

[7] D. B. Johnson and W. Zwaenepoel, "Recovery in distributed systems using optimistic message logging and checkpointing," *J. of Algorithms*, vol. 11, pp. 462–491, 1990.

[8] B. Randell, "System structure for software fault tolerance," *IEEE Trans. on Software Engineering*, vol. SE-1, pp. 220–232, June 1975.

[9] R. D. Schlichting and F. B. Schneider, "Fail-stop processors: An approach to designing fault-tolerant computing systems," *ACM Trans. on Computer Systems*, vol. 1, pp. 222–238, Aug. 1983.

[10] F. Cristian and F. Jahanian, "A timestamp-based checkpointing protocol for long-lived distributed computations," in *Proc. IEEE Symp. on Reliable Distributed Systems*, pp. 12–20, 1991.

[11] M. L. Powell and D. L. Presotto, "Publishing: A reliable broadcast communication mechanism," in *Proc. 9th ACM Symp. on Operating Systems Principles*, pp. 100–109, 1983.

[12] A. Borg, J. Baumbach, and S. Glazer, "A message system supporting fault-tolerance," in *Proc. 9th ACM Symp. on Operating Systems Principles*, pp. 90–99, 1983.

[13] A. Borg, W. Blau, W. Graetsch, F. Herrmann, and W. Oberle, "Fault tolerance under UNIX," *ACM Trans. on Computer Systems*, vol. 7, pp. 1–24, Feb. 1989.

[14] L. Lamport, "Time, clocks and the ordering of events in a distributed system," *Comm. of the ACM*, vol. 21, pp. 558–565, July 1978.

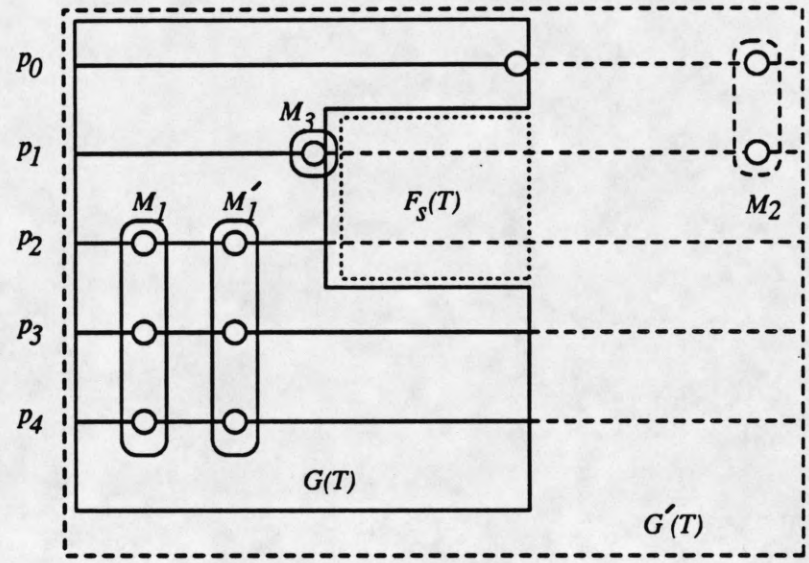
[15] K. P. Bogart, *Introductory combinatorics.* Pitman Publishing Inc., Massachusetts, 1983.

[16] I. Anderson, *Combinatorics of finite sets.* Clarendon Press, Oxford, 1987.

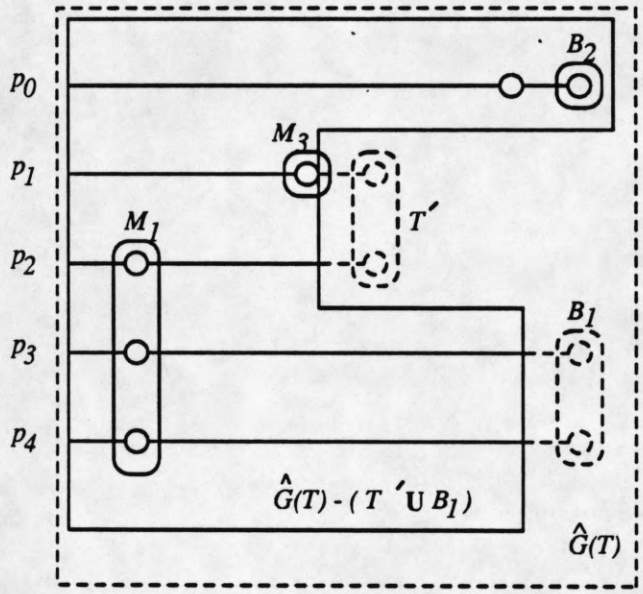
[17] R. P. Dilworth, "Some combinatorial problems on partially ordered sets," *Combinatorial Analysis, Proc. Symp. Appl. Math.*, vol. 10, pp. 85–90, 1960.

[18] D. B. West, "Personal communication." 1991.

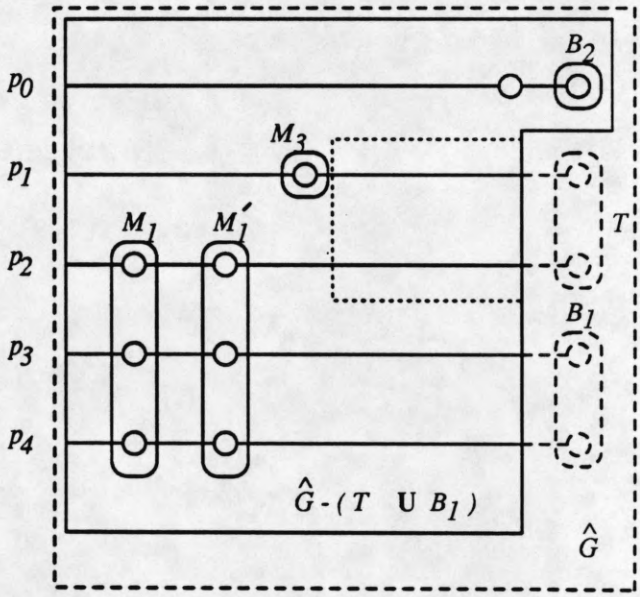
[19] W. Shu and L. V. Kalé, "Chare kernel - a runtime support system for parallel computations," *J. Parallel and Distributed Computing*, vol. 11, pp. 198–211, 1991.



(a)



(b)



(c)

Figure 11: Suppose (a) $M_1 \cup M_2$ forms the MM-chain of $G'(T)$. Then $M_1 \cup M_3 \cup B_2$ is an M-chain of $\hat{G}(T) - (T' \cup B_1)$ and forms the MM-chain of $\hat{G} - (T \cup B_1)$.